

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION FOR LETTERS PATENT

PASSWORD-BASED KEY MANAGEMENT

Inventor(s):

Mariusz H. Jakubowski

M. Kivanc Mihcak

ATTORNEY'S DOCKET NO. MS1-1664US

EL996276928

## **BACKGROUND**

Data security is vital to many individuals and business. This is particularly true in situations where the data is transmitted or stored in a digital form. Various methods have been devised that protect access to digital data. One such method involves protecting data using a password. Unfortunately, users typically select simple passwords that are easy to remember, thus making them relatively simple to discover using such methods as dictionary attacks.

Another common method for protecting data is to encrypt the data using an encryption key. Encryption keys can be quite complex, thus making them difficult to determine. However, complex encryption keys are very difficult, if not impossible, to remember. As such, complex encryption keys are typically stored on the user's computer for later use in accessing the protected data. Unfortunately, stored encryption keys are vulnerable to discovery by hackers. Additionally, in the case where either an encryption key or a password is sent electronically, the password or encryption may be intercepted in transit.

## **SUMMARY**

Implementations described and claimed herein address the foregoing problems by providing methods, systems, and data structures that permit data to be protected with complex keys, but which allow users to access the protected data using only a simple user id and password.

In accordance with one implementation, data is protected using a key-based forward transformation process. The password of each user that is authorized to access the data is then hashed to produce a hash value. A user key is then created for each user comprising an encrypted version of the master key, with the master

1 key being encrypted using the hash of the user's password as an encryption key.  
2 Each user's user key and user id are then associated in a user key data structure or  
3 database.

4 In accordance with another embodiment, when a user wishes to access the  
5 protected data, the user's user id is used to select the appropriate user key from the  
6 user key data structure. The user's password is then hashed to produce a hash  
7 value. This hash value is then used as a key to decrypt the user key to produce the  
8 master key. The protected data is then reverse transformed using the master key to  
9 produce the original data.

#### 10 11 **BRIEF DESCRIPTION OF THE DRAWINGS**

12 Fig. 1 illustrates an exemplary system for providing access to protected  
13 data.

14 Fig. 2 illustrates an exemplary implementation of the user key data  
15 structure generator shown in Fig. 1.

16 Fig. 3 illustrates an exemplary implementation of the data access module  
17 shown in Fig. 1.

18 Fig. 4 illustrates exemplary operations for producing a user key data  
19 structure.

20 Fig. 5 illustrates exemplary for accessing a master key using a user key data  
21 structure.

22 FIG. 6 illustrates an exemplary computer system for implementing  
23 embodiments of the systems and methods described herein.

## DETAILED DESCRIPTION

Described herein are exemplary systems, methods, and data structures for providing authorized users access to robustly protected data using only a user id and a user password. In accordance with various implementations described herein, data is protected by transforming the data using a key-based transformation process. The transformed data is then accessed using a key-based reverse transformation process that is complementary to the forward transformation process.

Turning first to Fig. 1, illustrated therein is an exemplary protected data access system 100. Included in the protected data access system 100 are a forward transformation module 110, a data access module 112, a user key data structure generator module 114, and a number of authorized users 116. Included in the data access module 112 are, among other things, a reverse transformation module 130 and a master key decryption module 128.

In general, the data access module 112 provides each of the authorized users 116 a mechanism by which they may access transformed data 122 using only their user ids and user passwords. As described in greater detail below, data 118 is transformed (e.g., encrypted, watermarked, or otherwise transformed or annotated) by the forward transformation module 110 using a master key 120. The transformed data 122 is then presented, or otherwise made available to, the data access module 112.

1 As also described in detail below, a user key data structure 126 is created by  
2 a user key data structure generator 114 using the same master key 120, as well as  
3 the user ids and user passwords of the authorized users 116. The user key data  
4 structure 126 includes, among other things, a uniquely encrypted form of the  
5 master key, called a user key, for each of the authorized users 116. The user key  
6 data structure 126 is then sent or delivered from the user key data structure  
7 generator 114 to the data access module 112.

8  
9 When a user wishes to access the data 118, the user 116 sends the user's id  
10 and password to the data access module 112. The user's id is then used by the  
11 master key decryption module 128 to access the user's user key in the user key  
12 data structure 126. The user's password is used to decrypt the user's user key. If  
13 the decryption of the user key is successful in producing the master key, the master  
14 key is then used by the reverse transformation module 130 to access (e.g., decrypt,  
15 verify, or other wise access using the master key) the protected data. The accessed  
16 data is then presented to the user 116.

17  
18 Having described the basic elements and operations of the protected data  
19 access system 100, a more detailed description of the various features and  
20 functions of the protected data access system 100 will now be provided. In  
21 accordance with one implementation, the data access module 112, the forward  
22 transformation module 110, user key data structure generation module 114, as well  
23 as the various modules included therein, are composed of computer executable  
24 instructions that are stored or embodied in one or more types of computer-readable  
25

1 media. As used herein, computer-readable media may be any available media that  
2 can store and/or embody computer executable instructions and that may be  
3 accessed by a computing system or computing process. Computer-readable-media  
4 may include, without limitation, both volatile and nonvolatile media, removable  
5 and non-removable media, and modulated data signals. The term “modulated data  
6 signal” refers to a signal that has one or more of its characteristics set or changed  
7 in such a manner as to encode information in the signal.

8  
9 Generally, the modules 110, 112, and 114, and the various modules  
10 included therein, may include various routines, programs, objects, components,  
11 data structures, etc., that perform particular tasks or operations or implement  
12 particular abstract data types. For example, in accordance with one  
13 implementation, the user key data structure generation module 114 performs the  
14 operations illustrated in Fig. 3 and creates the user key data structure 226  
15 illustrated in Fig. 2. Similarly, in accordance with one implementation, the data  
16 access module 112 performs the operations illustrated in Fig. 5.

17  
18 It should be understood that while the modules 110, 112, and 114, and the  
19 various modules included therein, are described herein as comprising computer  
20 executable instructions embodied in computer-readable media, the modules 110,  
21 112, and 114, the modules included therein, and any or all of the functions or  
22 operations performed thereby, may likewise be embodied all or in part as  
23 interconnected machine logic circuits or circuit modules within a computing  
24 device. Stated another way, it is contemplated that the program modules 110, 112,  
25

1 and 114, the modules included therein, and their operations and functions, such as  
2 the operations shown and described with respect to Figs. 3 and 5, may be  
3 implemented as hardware, software, firmware, or various combinations of  
4 hardware, software, and firmware. The implementation is a matter of choice  
5 dependent on performance requirements of the data access system 100.

6 Any of the modules 110, 112, and 114, or the modules included therein,  
7 may be executed or implemented in a single computing device or in a distributed  
8 computing environment, where tasks are performed by remote processing devices  
9 or systems that are linked through a communications network. In accordance with  
10 one implementation, the forward transformation module 110, the data access  
11 module 112, and the user key data structure generator module 114 are each  
12 implemented in or by separate computing devices. Likewise, in accordance with  
13 one implementation, each of the users 116 accesses the data-access module 112  
14 from one or more separate computing devices.  
15

16 As shown and described with respect to Fig. 1, the forward transformation  
17 module 110, user key data structure generation module 114, and the authorized  
18 users 116 each interact or communicate in some manner with the data access  
19 module 112. The precise manner in which these interactions take place may vary,  
20 depending on the manner in which the individual elements and the protected data  
21 access system 100 as a whole are implemented, and/or the purpose of the  
22 communication.  
23  
24  
25

1 For example, in accordance with one implementation, the forward  
2 transformation module 110 is connected to the data access module 112 via a  
3 network, such as an intranet or the Internet. In this implementation, the forward  
4 transformation module 110 sends the transformed data 122 to the data access  
5 module 112 via the network. In this implementation, the transformed data 122 may  
6 be sent to the data access module 112 using any number of communication  
7 protocols, either proprietary or non-proprietary.

8  
9 Similarly, in accordance with one implementation, the user key data  
10 structure generator module 114 may also be connected to the data access module  
11 112 via a network, such as an intranet or the Internet. In this implementation, user  
12 key data structure generator module 114 sends the user key data structure 126 to  
13 the data access module 112 via the network. However, as described below, in  
14 accordance with one embodiment, the user key data structure 126 is not delivered  
15 to the data access module using a network connection. Rather, for security  
16 purposes, the user key data structure 126 is delivered to the data access module  
17 112 "off-line" using a removable media, such as a floppy disk, CD-ROM, or the  
18 like.

19  
20 As noted, each of the authorized users 116 communicates with the data  
21 access module to send user Ids and passwords, and to access the data 118. In  
22 accordance with one embodiment, users may 116 communicate with the data  
23 access module 112 using one or more separate computing devices or processes that  
24 are remote from the data access module 112. That is, the authorized users may  
25



1 remotely communicate with the data access module 112. For example, one or more  
2 of the users may remotely communicate with the data access module 112 using a  
3 personal computer connected to the data access module 112 via a network, such as  
4 an intranet or the Internet. In accordance with another embodiment, one or more of  
5 the authorized users may have direct access to a computer that is executing the  
6 data access module 112. That is, one or more users may directly communicate  
7 with the data access module 112. In other embodiments, some authorized users  
8 may remotely communicate with the data access module 112, while other users  
9 may directly communicate with the access module.  
10

11 With respect to the forward transformation module 110, as shown, the  
12 forward transformation module 110 receives data 118, and protects the data 118  
13 using a master key (MK) 120. The data 118 may have any of number of forms, and  
14 may comprise various types of information. The master key 120 may be of any  
15 size and/or type that is compatible with the transformation techniques used by the  
16 forward transformation module 110 and the reverse transformation module 130.  
17 Furthermore, the master key 120 may be produced or obtained from any of a  
18 number of methods or sources. However, it is preferable that the master key 120  
19 be of a size, type, and/or produced by a process that makes the likelihood of  
20 discovery of the master key 120 statistically insignificant. For example, and  
21 without limitation, in accordance with one implementation, the master key 120  
22 may be generated as an output of a secure random number generator. In  
23  
24  
25

1 accordance with another embodiment, the master key may be generated as a hash  
2 value of text or other information.

3 In general, the forward transformation module 110 uses the master key 120  
4 in some manner to transform or annotate the data 118 to produce the transformed  
5 data 122. As will described in greater detail below, the reverse transformation  
6 module 130 of the data access module 112 then uses the master key to either  
7 reverse the transformation performed by the forward transformation module 110,  
8 or to verify the protected data.

9  
10 For example, in accordance with one implementation, herein called the  
11 encryption implementation, the forward transformation module 110 encrypts the  
12 data 118 to produce the transformed data 122. In accordance with one  
13 implementation, herein called the watermarking implementation, the forward  
14 transformation module 110 watermarks the data 118 to produce the transformed  
15 data 122. In accordance with other implementations, the forward transformation  
16 module 110 uses the master key to transformation or annotate the data 118 in other  
17 manners to produce the transformed data 122.

18  
19 In accordance with the data encryption implementation, the forward  
20 transformation module 110 encrypts the data 118 to produce transformed data 122.  
21 In accordance with this implementation, the forward transformation module 110  
22 uses an encryption process that is symmetrical with a decryption process used by  
23 the reverse transformation module 130 in the data access module 112. That is, the  
24 master key 120 that is used by the forward transformation module 110 to produce  
25

1 the transformed data 122 is the same master key that is used by the reverse  
2 transformation module 130 to decrypting the transformed data 122.

3 In accordance with this encryption implementation, any of a number of  
4 symmetrical data encryption/decryption techniques may be used by the forward  
5 transformation module 110 and the reverse transformation module 130. For  
6 example, and without limitation, the forward transformation module 110 may use,  
7 without limitation, a commonly accepted stream cipher (e.g., RC4) or block cipher  
8 (e.g., 3DES or AES).  
9

10 In accordance with the watermarking implementation, the forward  
11 transformation module 110 watermarks the data 118 using the master key to  
12 produce the transformed data 122. That is, in accordance with this watermarking  
13 implementation, the forward transformation module 110 imbeds the master key as  
14 watermark in the data 118 to produce the transformed data 122. Any number of  
15 public-key, private-key, or detection-key type watermarking techniques may be  
16 used in accordance with this watermarking implementation. For example, and  
17 without limitation, in accordance with one implementation, a wavelet-based  
18 spread-spectrum type watermarking technique is used by the forward  
19 transformation module 110 to form the transformed data 122.  
20

21 After the data has been 118 transformed by the forward transformation  
22 module 110, the resulting transformed data 122 is made available to the data  
23 access module 112. In accordance with one embodiment, the transformed data 122  
24 is sent to the data access module 112, where it is stored for later access by a user  
25

1 116. In accordance with another embodiment, the transformed data 122 is sent to  
2 the data access module 112 only when it is requested by a user 116. In such a case,  
3 a user 116 sends a request for the transformed data 122 to the data reverse  
4 transformation module 130, which in turn sends a request for the transformed data  
5 122 to the forward transformation module 110. The forward transformation  
6 module 110 then sends the transformed data 122 to the data access module 112 for  
7 processing and presentation to the user 116. The manner in which the transformed  
8 data is presented by the data access module 112 is described in detail below with  
9 respect to Fig. 3.  
10

11 Turning now to Fig. 2, illustrated there are further details an exemplary user  
12 key data structure generator module 114. As shown, the user key data structure  
13 generator module 114 includes a hashing module 210, a master key encryption and  
14 integrity module 212, and a user key data structure creation module 216. In  
15 operation, the user key data structure generator module 114 receives as input user  
16 passwords 222, the master key 120, user ids 224, and produces as an output the  
17 user key data structure 126. In accordance with one embodiment, each of the user  
18 ids received by the user key data structure generator module 114 is a user id of an  
19 authorized user 116. Likewise, each of the user passwords received by the user key  
20 data structure generator module 114 is a password of an authorized user 116. The  
21 master key 120 received by the user key data structure generator module 114 is  
22 identical to the previously described master key 120 received by the forward  
23 transformation module 110.  
24  
25

1 In accordance with one implementation, user ids and user passwords are  
2 selected by the users themselves and presented to the user key data structure  
3 generator module 114 via a secure communications channel, or other secure  
4 mechanism. In accordance with another implementation, the user ids and user  
5 passwords are selected by the user key data structure generator module 114 and  
6 transmitted to the appropriate users via a secure communications channel, or other  
7 secure mechanism.

8  
9 In general, the hashing module 210 receives as an input a user password  
10 and 220 and produces as an output a hash value ( $H_i$ ). In accordance with one  
11 implementation, the hashing module 210 employs a one-way hash function to  
12 produce the hash value from the password. As will be appreciated to those skilled  
13 in the art, a one-way hash function is a mathematical function that takes as an  
14 input a variable-length string and converts the variable length string into a fixed-  
15 length binary sequence. Often the length of output of the hash function is much  
16 less than the length of the input. One-way hash functions are typically designed  
17 such that it is extremely improbable that the input string can be determined from  
18 the output binary sequence. That is, it is extremely difficult to find an input string  
19 that maps to a given output sequence. Furthermore, a well-designed hash function  
20 bears the property of low or insignificant collision probability (i.e., the probability  
21 of two different inputs' yielding the same hash value). Some examples from the  
22 literature are MD-5 and SHA-1 hash functions.  
23  
24  
25

1 In accordance with another implementation, the hashing module 210  
2 employs a cryptographic hash function to produce the hash value from the  
3 password. As will be appreciated, a cryptographic hash function is a mathematical  
4 function that is both one-way and collision-resistant. A hash function is collision-  
5 resistant if it is extremely improbable to find any two distinct input strings that  
6 map to the same output sequence.

7 As shown in Fig. 2, the hash value ( $H_i$ ) is received by the master key  
8 encryption and integrity module 212. Additionally, the master key encryption and  
9 integrity module 212 receives the master key. In general, the master key  
10 encryption and integrity module 212 encrypts the master key using the hash value  
11 ( $H_i$ ) as an encryption key, to produce an encrypted master key. In accordance with  
12 one implementation, the master key encryption and integrity module 212 uses an  
13 encryption process that is symmetrical with the decryption process used by the  
14 master key decryption module 128 in the data access module 112. That is, the  
15 encryption key (hash value ( $H_i$ )) that is used by the master key encryption and  
16 integrity module 212 to produce the encrypted master key is the same decryption  
17 key that is used by the master key decryption module 128 to decrypt the encrypted  
18 master key.  
19  
20

21 Any of a number of symmetrical data encryption/decryption techniques  
22 may be used by the master key encryption and integrity module 212 and the  
23 master key decryption module 128. For example, and without limitation, the  
24  
25

1 master key encryption and integrity module 212 may use a block cipher such as  
2 3DES or AES or a stream cipher such as RC4.

3 In accordance with one implementation, the encoded master key is then  
4 specified as the user key ( $UK_i$ ) for the user whose password was input to the  
5 hashing module to produce the encoding key used to encode the master key. This  
6 user key ( $UK_i$ ) is then sent to the user key data structure creation module 216.  
7 However, in accordance with another implementation, the encoded master key is  
8 further processed by the master key encryption and integrity module 212 before it  
9 is sent to the user key data structure creation module 216  
10

11 In accordance with one implementation, in addition to encrypting the  
12 master key 120, the master key encryption and integrity module 212 also adds an  
13 optional data integrity verification feature to the encrypted master key. For  
14 example, in accordance with one implementation, the master key encryption and  
15 integrity module 212 adds a checksum or message authentication code to the  
16 encrypted master key. In accordance with one implementation, the master key  
17 encryption and integrity module 212 uses the hash value ( $H_i$ ) produced by the hash  
18 function to produce a keyed-hash message authentication code (HMAC).  
19

20 In the case where a data integrity verification feature is added to the  
21 encrypted master key, the encoded master key, including the data integrity  
22 verification feature, is then specified as the user key ( $UK_i$ ) for the user whose  
23 password was input to the hashing module to produce the encoding key used to  
24 encode the master key.  
25

1 As shown in Fig. 2, the user key ( $UK_i$ ) is received by the table creation  
2 module 216. Additionally, the user key data structure creation module 216 receives  
3 the user Id ( $Id_i$ ) corresponding to the user whose password was used as an input to  
4 the hashing module 210. The user key data structure creation module 216 then  
5 associates the user key ( $UK_i$ ) with the user Id ( $Id_i$ ) to produce a “user Id-user key  
6 pair.”

7 Typically a hash value ( $H_i$ ), user key ( $UK_i$ ), and user Id-user key pair will  
8 be created for each authorized user 116. Each of the Id-user key pairs will then be  
9 combined by the user key data structure creation module 216 to form the user key  
10 data structure 120. The user key data structure 120 may have various forms. For  
11 example, and without limitation, the user key data structure 120 may comprise a  
12 table, such as shown in Fig. 2. However, those skilled in the art will appreciate that  
13 the user id and the user key may be associated in various other ways and various  
14 other types of data structures.  
15

16 Turning now to Fig. 3, illustrated therein are various exemplary operations  
17 that may be performed in a process for generating the user key data structure 126.  
18 In accordance with one implementation, the operations 300 are performed by the  
19 user key data structure generator module 114. In accordance with other  
20 implementations, the operations may be performed by other modules or systems.  
21

22 At the beginning of the process, a receive operation 302 obtains a user Id  
23 and associated user password for a given authorized user. Next, a hashing  
24 operation 304 hashes the given user’s password to create a hash value. An  
25



1 encryption operation 306 then produce a user key by encrypting the master key  
2 using the hash value produced in operation 304.

3       Following the encryption operation 306, a creation operation 308 then  
4 creates a user id – user key pair by associating the user key created by the  
5 encryption operation 306 with the user id received in receive operation 302. Next a  
6 determination operation 310 determines if a user id – user key pair has been  
7 created for each authorized user. If a user id – user key pair has not been created  
8 for each authorized user, the process proceeds back to the receive operation 302,  
9 and the operations 302, 304, 306, 308, and 310 are repeated for each authorized  
10 user. If a user id – user key pair has been created for each authorized user, a  
11 combination operation combines each of the user id – user key pairs in user key  
12 data structure, such as user key data structure 126, or the like.  
13

14       Turning now to Fig. 4, illustrated therein are details of an exemplary data  
15 access module 112. As shown, the data access module 112 includes a master key  
16 decryption module 128, a reverse transformation module 130, and an error handler  
17 module 410. The master key decryption module 128 includes a hashing module  
18 410 and a user key decryption and integrity module 412. In general, the data  
19 access module functions as an access point for authorized users 116 to access data  
20 encrypted with a complex and/or lengthy master key, using only their user ids and  
21 password.  
22

23       In operation, the data access module 112 receives a user id 222 and a user  
24 password 224 for an authorized user 116. The hashing module 410 receives the  
25

1 user password and produces as an output a hash value ( $H_i$ ). In accordance with one  
2 implementation, the hashing module 410 uses a hashing function that is identical  
3 to the hashing function used by the hashing module 210, described above with  
4 respect to Fig.2.

5 The user key decryption and integrity module 412 receives the user id 224  
6 and the hash value ( $H_i$ ) output from the hashing module 410. The user key  
7 decryption and integrity module 412 then uses the user id to retrieve from the user  
8 key data structure 126 a user key ( $UK_i$ ) corresponding to the received user id. That  
9 is, the user key decryption and integrity module 412 retrieves from the user key  
10 data structure 126 the user key ( $UK_i$ ) associated with the user id in a use id/user  
11 key pair.  
12

13 In the case where the user key ( $UK_i$ ) was originally formed without a data  
14 integrity verification feature, the user key decryption and integrity module 412  
15 attempts to decrypt the retrieved user key ( $UK_i$ ) using the output of the hashing  
16 module 410 as a decryption key. As previously noted, the decryption algorithm  
17 used by the user key decryption and integrity module 412 will preferably be  
18 reciprocal to the encryption algorithm used by the master key encryption and  
19 integrity module 212. As such, if the user inputs the proper user id and password,  
20 the user key will be decrypted to form the original master key.  
21

22 In the case where the user key ( $UK_i$ ) was originally formed with a data  
23 integrity verification feature, the user key decryption and integrity module 412  
24 first attempts to verify the integrity of the user key. The user key decryption and  
25

1 integrity module 412 will attempt to verify the integrity of the user key using the  
2 data integrity verification feature added to the user key by the master key  
3 encryption and integrity module 212. If the integrity of the user key is verified, the  
4 user key decryption and integrity module 412 will attempt to decrypt the retrieved  
5 user key (UK<sub>i</sub>) using the output of the hashing module 410 as a decryption key, as  
6 described above. Alternately, the encrypted user key may be first decrypted, and  
7 the integrity-checking mechanism then verifies that the decrypted key is correct.  
8

9 In accordance with one implementation, if the decryption of the user key  
10 fails, the error handler module 416 is notified of the failure. The error handler  
11 module 416 may take any number of actions in response to such a failure. For  
12 example, and without limitation, in accordance with one implementation, the error  
13 handler module 416 informs the user that an error has occurred. In accordance  
14 with another implementation, the error handler module 416 keeps track of  
15 unsuccessful attempts by a user to access data, and blocks the user from accessing  
16 data for a predetermined time period if the user exceeds a predetermined number  
17 of failed data access attempts. In accordance with yet another embodiment, the  
18 error handler module 416 reports information regarding failed data access attempts  
19 to a system administrator. In accordance with another embodiment, the error  
20 handler module 416 waits a progressively increasing amount of time after a failed  
21 attempt by a user to access data before allowing the user to attempt to access the  
22 data. In accordance with another embodiment, the error handler module 416  
23  
24  
25

1 deletes a user id and associated user password from the user key data structure  
2 after a predetermined number of failed attempts to access the data.

3 In the case where the user key authentication and decryption module 412  
4 successfully decrypts the master key, the reverse transformation module 130 then  
5 decrypts the transformed data 122. The decryption algorithm used by the reverse  
6 transformation module 130 will preferably be reciprocal to the encryption  
7 algorithm used by the forward transformation module 110, described above with  
8 respect to Fig. 1. The decrypted data 118 may then either be delivered or presented  
9 to the authorized user 116 whose user id and password were used by the data  
10 access module 112 to decrypt the data.  
11

12 Turning now to Fig. 5, illustrated therein are various exemplary operations  
13 500 that may be performed in a process for decrypting data encrypted with a  
14 master key. In accordance with one implementation, the operations 500 are  
15 performed by the data access module. In accordance with other embodiment, the  
16 operations 500 may be performed by other modules or systems.  
17

18 At the beginning of the process, a receive operation 502 receives or obtains  
19 a user id and associated user password for a given authorized user. A hashing  
20 operation 504 then hashes the authorized user's password to create a hash value. A  
21 retrieve operation 506 then uses the user id to retrieve a user key comprising an  
22 encrypted version of the master key that was used to encrypt the data.  
23

24 In accordance with one implementation, the user key is retrieved from a  
25 data structure including a plurality of user ids, each of which is associated with a

1 unique user key. For example, in accordance with one implementation the data  
2 structure from which the user key is retrieved may comprise the user key data  
3 structure 120 described above.

4 In accordance with one implementation, the user key includes an integrity  
5 verification feature. In such a case, following the retrieve operation 506, a verify  
6 operation 508 verifies the integrity of the user key. If the verify operation 508 does  
7 not verify the integrity of the user key, an error handling operation 510 is  
8 performed. The error handling operation 510 may perform various error handling  
9 actions, such as, without limitation, the actions described above with respect to the  
10 error handling module 416.  
11

12 In accordance with another implementation, the user key does not include  
13 an integrity verification feature, or an integrity verification feature is ignored in  
14 the operations 500. In such a case, the verify operation 508 is not performed.  
15 Rather, a decrypt operation 512 is performed following the retrieve operation 506.  
16 The decrypt operation 512 attempts to decrypt the user key to produce the master  
17 key, using the hash value produced during the hashing operation 504.  
18

19 Next, an optional determination operation 514 determines if the decrypt  
20 operation 512 was successful in decrypting the user key to produce the master key.  
21 If the determination operation 514 determines that the decrypt operation 512 was  
22 not successful in decrypting the user key to produce the master key, the error  
23 handling operation 510 is performed, as described above. If, however, the  
24 determination operation 514 determines that the decrypt operation 512 was  
25

1 successful in decrypting the user key to produce the master key, the master key, a  
2 data presentation operation 516 then presents the transformed data using the  
3 master key.

4 Following the decrypt operation 516 a data presentation operation 518  
5 presents the decrypted data to the user. In accordance with one implementation,  
6 the data presentation operation 518 delivers the decrypted data to the user. In  
7 accordance with another implementation, the data presentation operation 518  
8 delivers the decrypted data to the user.

9  
10 FIG. 6 illustrates one exemplary computing environment 610 in which the  
11 various systems, methods, and data structures described herein may be  
12 implemented. The exemplary computing environment 610 is only one example of  
13 a suitable computing environment and is not intended to suggest any limitation as  
14 to the scope of use or functionality of the systems, methods, and data structures  
15 described herein. Neither should computing environment 610 be interpreted as  
16 having any dependency or requirement relating to any one or combination of  
17 components illustrated in computing environment 610.

18  
19 The systems, methods, and data structures described herein are operational  
20 with numerous other general purpose or special purpose computing system  
21 environments or configurations. Examples of well known computing systems,  
22 environments, and/or configurations that may be suitable include, but are not  
23 limited to, personal computers, server computers, thin clients, thick clients, hand-  
24 held or laptop devices, multiprocessor systems, microprocessor-based systems, set  
25

1 top boxes, programmable consumer electronics, network PCs, minicomputers,  
2 mainframe computers, distributed computing environments that include any of the  
3 above systems or devices, and the like.

4 The exemplary operating environment 610 of FIG. 6 includes a general  
5 purpose computing device in the form of a computer 620, including a processing  
6 unit 621, a system memory 622, and a system bus 623 that operatively couples  
7 various system components include the system memory to the processing unit 621.  
8 There may be only one or there may be more than one processing unit 621, such  
9 that the processor of computer 620 comprises a single central-processing unit  
10 (CPU), or a plurality of processing units, commonly referred to as a parallel  
11 processing environment. The computer 620 may be a conventional computer, a  
12 distributed computer, or any other type of computer.

14 The system bus 623 may be any of several types of bus structures including  
15 a memory bus or memory controller, a peripheral bus, and a local bus using any of  
16 a variety of bus architectures. The system memory may also be referred to as  
17 simply the memory, and includes read only memory (ROM) 624 and random  
18 access memory (RAM) 625. A basic input/output system (BIOS) 626, containing  
19 the basic routines that help to transfer information between elements within the  
20 computer 620, such as during start-up, is stored in ROM 624. The computer 620  
21 may further includes a hard disk drive interface 627 for reading from and writing  
22 to a hard disk, not shown, a magnetic disk drive 628 for reading from or writing to  
23  
24  
25

1 a removable magnetic disk 629, and an optical disk drive 630 for reading from or  
2 writing to a removable optical disk 631 such as a CD ROM or other optical media.

3 The hard disk drive 627, magnetic disk drive 628, and optical disk drive  
4 630 are connected to the system bus 623 by a hard disk drive interface 632, a  
5 magnetic disk drive interface 633, and an optical disk drive interface 634,  
6 respectively. The drives and their associated computer-readable media provide  
7 nonvolatile storage of computer-readable instructions, data structures, program  
8 modules and other data for the computer 620. It should be appreciated by those  
9 skilled in the art that any type of computer-readable media which can store data  
10 that is accessible by a computer, such as magnetic cassettes, flash memory cards,  
11 digital video disks, Bernoulli cartridges, random access memories (RAMs), read  
12 only memories (ROMs), and the like, may be used in the exemplary operating  
13 environment.  
14

15 A number of program modules may be stored on the hard disk, magnetic  
16 disk 629, optical disk 631, ROM 624, or RAM 625, including an operating system  
17 635, one or more application programs 636, other program modules 637, and  
18 program data 638. A user may enter commands and information into the personal  
19 computer 620 through input devices such as a keyboard 40 and pointing device  
20 642. Other input devices (not shown) may include a microphone, joystick, game  
21 pad, satellite dish, scanner, or the like. These and other input devices are often  
22 connected to the processing unit 621 through a serial port interface 646 that is  
23 coupled to the system bus, but may be connected by other interfaces, such as a  
24  
25



1 parallel port, game port, or a universal serial bus (USB). A monitor 647 or other  
2 type of display device is also connected to the system bus 623 via an interface,  
3 such as a video adapter 648. In addition to the monitor, computers typically  
4 include other peripheral output devices (not shown), such as speakers and printers.

5 The computer 620 may operate in a networked environment using logical  
6 connections to one or more remote computers, such as remote computer 649.  
7 These logical connections may be achieved by a communication device coupled to  
8 or a part of the computer 620, or in other manners. The remote computer 649 may  
9 be another computer, a server, a router, a network PC, a client, a peer device or  
10 other common network node, and typically includes many or all of the elements  
11 described above relative to the computer 620, although only a memory storage  
12 device 650 has been illustrated in FIG. 6. The logical connections depicted in FIG.  
13 6 include a local-area network (LAN) 651 and a wide-area network (WAN) 652.  
14 Such networking environments are commonplace in office networks, enterprise-  
15 wide computer networks, intranets and the Internet, which are all types of  
16 networks.  
17  
18

19 When used in a LAN-networking environment, the computer 620 is  
20 connected to the local network 651 through a network interface or adapter 653,  
21 which is one type of communications device. When used in a WAN-networking  
22 environment, the computer 620 typically includes a modem 654, a type of  
23 communications device, or any other type of communications device for  
24 establishing communications over the wide area network 652. The modem 654,  
25

1 which may be internal or external, is connected to the system bus 623 via the serial  
2 port interface 646. In a networked environment, program modules depicted  
3 relative to the personal computer 620, or portions thereof, may be stored in the  
4 remote memory storage device. It is appreciated that the network connections  
5 shown are exemplary and other means of and communications devices for  
6 establishing a communications link between the computers may be used.

7  
8 Although some exemplary methods, systems, and data structures have been  
9 illustrated in the accompanying drawings and described in the foregoing Detailed  
10 Description, it will be understood that the methods, systems, and data structures  
11 shown and described are not limited to the exemplary embodiments and  
12 implementations described, but are capable of numerous rearrangements,  
13 modifications and substitutions without departing from the spirit set forth and  
14 defined by the following claims.  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25